# 論文 / 著書情報
# Article / Book Information

| | |
|---|---|
| Title | A File Recommendation Method Based on Task Workflow Patterns Using File-access Logs |
| Author | Qiang Song, Takayuki Kawabata, Fumiaki Itoh, Yousuke Watanabe, Haruo Yokota |
| Journal/Book name | Lecture notes in computer science, LNCS, 8056, , pp. 410-417 |
| 発行日 / Issue date | 2013, 8 |
| DOI | http://dx.doi.org/10.1007/978-3-642-40173-2_33 |
| 権利情報 / Copyright | The original publication is available at www.springerlink.com. |
| Note | このファイルは著者（最終）版です。<br>This file is author (final) version. |

# A File Recommendation Method Based on Task Workflow Patterns Using File-access Logs

Qiang Song[1], Takayuki Kawabata[2], Fumiaki Itoh[2], Yousuke Watanabe[1], and Haruo Yokota[1]

[1] Tokyo Institute of Technology
[2] Canon Inc.
{soukyou,watanabe}@de.cs.titech.ac.jp,{kawabata.takayuki,ito.fumiaki}@canon.co.jp,yokota@cs.titech.ac.jp

**Abstract.** In recent years, office workers spend much time and effort searching for the documents required for their jobs. To reduce these costs, we propose a new method for recommending files and operations on them. Existing technologies for recommendation, such as collaborative filtering, suffer from two problems. First, they can only work with documents that have been accessed in the past, so that they cannot recommend when only newly generated documents are inputted. Second, they cannot easily handle sequences involving similar or differently ordered elements because of the strict matching used in the access sequences. To solve these problems, such minor variations should be ignored. In our proposed method, we introduce the concepts of abstract files as groups of similar files used for a similar purpose, abstract tasks as groups of similar tasks, and frequent abstract workflows grouped from similar workflows, which are sequences of abstract tasks. In experiments using real file-access logs, we confirmed that our proposed method could extract workflow patterns with longer sequences and higher support-count values, which are more suitable as recommendations.

**Keywords:** File recommendation, File abstraction, Abstract task, Abstract workflow, Log analysis

## 1 Introduction

The numbers of files in file systems have grown dramatically. As a consequence of this increase, office workers now spend much time and effort searching for files that include documents and data required for their business [1]. It is therefore important to find methods for reducing the time wasted in ineffective and unproductive searching. Also, in office, the searching targets are not limited to documents, the abstract working processes which we call them abstract workflows in this paper are also being searched by users.

Collaborative filtering is well known as an algorithm for making recommendations. However, it only works with files accessed in the past. Recommendation systems based on this algorithm do not work well on newly generated files. Moreover, collaborative filtering does not handle well sequences that include elements

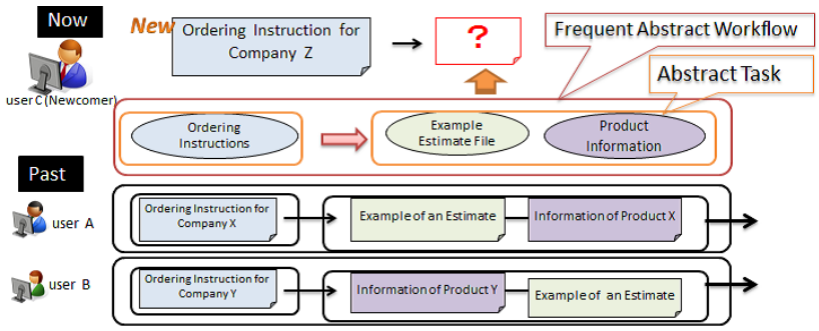


**Fig. 1.** Our Challenges in File Recommendation

that are similar but not identical, or that have a different ordering of elements, because the algorithm uses strict matching of elements in access sequences.

In this paper, we propose a method for recommending files and types of operations on them (e.g. open, close), that supports creative business processes as an operational tool. We consider the abstract workflows for a user. These are different from the ordinary workflows described by users, being generated by mining the file-access histories of the users. To obtain good recommendations, it is important to ignore small variations in the matching process. In this paper, we introduce for the first time the concepts of abstract files, abstract tasks, and frequent abstract workflows. The differences between collaborative filtering and our proposed method in this paper are that we add the concept of tasks to weaken the constraint of strict matching for access sequences and we apply abstraction to files, tasks, and workflows to handle newly generated files. The evaluation results indicate that our method extracts workflow patterns with longer sequences and higher support-count values (occurrence times of workflow patterns), which are more suitable as recommendations. Moreover, the results demonstrate that using the concepts of abstract tasks and abstract workflows improves the quality of the recommendations.

As a previous work, we compared several file similarity calculation methods [5]. In this paper, we focus on how to abstract tasks and workflows for recommendation.

## 2 Goal and Approach

Figure 1 illustrates an example of the type of workflow pattern we are investigating. We assume that user $C$ creates a new file. The goal is to predict the files required, for recommendation to user $C$.

In our approach, we search for similar patterns of operation in logs and make recommendations based on them. For example, in the two patterns of user $A$ and $B$, different files were accessed in different orders, but it is considered that users $A$ and $B$ were doing essentially the same work. Therefore, we expect

to extract an abstract pattern of work. Using this abstract workflow pattern, it becomes possible to recommend files such as "Example of an estimate" or "product information file" to user $C$.

While users access files located on a file server, file-access histories are stored in a log file. Each record in the log file includes timestamp, user, operation and filename information. Our method extracts abstract tasks and workflows from the log file, and reserves them in a database. The method then monitors the current file accesses by a user, and searches for workflows in the database that match the access patterns of that user to infer the user's current workflow pattern. Based on the inferred workflow, the method recommends those files, together with operations on them.

## 3 Definition of Workflow Model

In general, the orders of sequences of individual operations will vary, even if the outlines of the workflows are the same. To ignore such subtle differences in operational order in similar workflows, we introduce the concept of an abstract task as a group of similar combinations of file and operation, and an abstract workflow as a sequence of abstract tasks.

**[Task]:** A *task* is a subsequence of records in the log file. It is the basic unit in a working process's workflow. For example, from the log below, we set TGBT=3 minutes.

- Record 1: 12:00 [Open File A]
- Record 2: 12:01 [Open File B]
- Record 3: 12:10 [Open File C]
- Record 4: 13:00 [Open File D]

Three tasks will be extracted from these four records.

- Task 1: {[Open File A], [Open File B]}
- Task 2: {[Open File C]}
- Task 3: {[Open File D]}

**[Abstract Task]:** An *abstract task* is a set of combinations of file and operation derived by clustering similar tasks, in order to ignore small variations in between tasks. In the above example, three abstract tasks would be derived as follows.

- Abstract Task 1: [Open File-Cluster 1], [Open File-Cluster 2]
- Abstract Task 2: [Open File-Cluster 1]
- Abstract Task 3: [Open File-Cluster 3]

**[Abstract Workflow]:** An *abstract workflow* is a sequence of abstract tasks. For the above example, we set TGBW=30 minutes and can extract two abstract workflows that are sequences of the abstract tasks.

- Abstract Workflow 1: [Abstract Task 1] → [Abstract Task 2]
- Abstract Workflow 2: [Abstract Task 3]

**[Frequent Abstract Workflow]:** *Frequent abstract workflows* are groups of similar workflows, in which sequences of abstract tasks appear frequently.

## 4 Proposed method

The process flow comprises two parts, namely the offline and online parts.

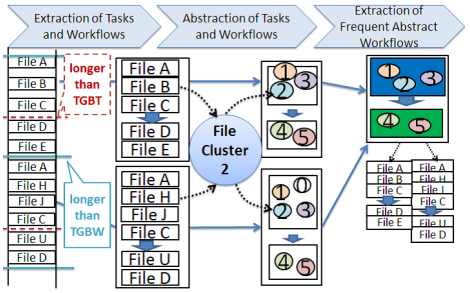


**Fig. 2.** Offline Part

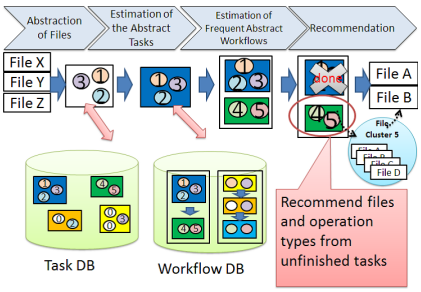


**Fig. 3.** Online Part

### 4.1 Offline Part

The aim of offline part is to extract abstract tasks and frequent abstract workflows from file-access logs.

**[Extraction of Tasks and Workflow]:** We first partition the log file in terms of user ID. If there are no file operations by a user during an interval longer than a certain time (TGBT for task, TGBW for workflow), we infer that the previous task/workflow had finished before the interval and the next task/workflow has started after it. We cut out tasks by parameter TGBT (150 seconds in experient) and workflows by parameter TGBW (1800 seconds in experient). In Figure 2, We partition the log into four tasks and two workflows.

**[Abstraction of Tasks and Workflows]:** To treat files with small differences as the same work, we want to cluster similar files used for the same purposes together. Instead of simply using the contents of the files, we calculate files' similarity based on the copy relationship and the similarity in filenames. Using the similarity between files, we apply agglomerative hierarchical clustering [2] to the files. Abstraction of files means replacing files, which appeared in the access log, with the corresponding clusters. We calculate the similarity between tasks by using the similarity between files. The similarity between tasks is a metric representing the degree of matching of two tasks in terms of file operations. It is large when two tasks have numerous abstract file operations in common. Here, we use Dice's Coefficient to calculate the degree of similarity. The formula for the similarity between $TaskA$ and $TaskB$ is

$$sim(TaskA, TaskB) = \frac{2|TaskA \cap TaskB|}{|TaskA| + |TaskB|}. \quad (1)$$

Based on the degree of similarity between tasks, we group tasks with a high degree of similarity together in a cluster as abstract tasks. We set two thresholds to filter out small clusters and unnecessary items inside clusters. First, only when a cluster contains more than Minimum Number of Tasks (MNT, 2 in experiment) tasks, will it be treated as an abstract task. Second, only the files inside a cluster, which occurrences ratio in tasks is more than a Minimum Emergence Ratio (MER, 0.5 in experiment), will be included in the abstract task. After the task abstraction, we simply replace each task in a workflow with the corresponding abstract task to obtain the abstract workflow.

**[Extraction of Frequent Abstract Workflows]:** To remove any infrequent abstract workflows created, we extract those that appear frequently as frequent abstract workflows. Two parameters are used, namely the Minimum Occurrence Time (MOT, 2 in experiment) and the Minimum Sequence Length (MSL, 2 in experiment) of the workflow sequence.

### 4.2 Online Part

The aim of online part is to find matching abstract tasks and frequent abstract workflows in database while monitors user's current operation, and then recommends files and operations on them to the user. As mentioned in Section 3, even when users are doing the same type of work, different files are usually being handled. For this reason, we abstract files being operated on currently by the user. We then estimate the abstract task and the frequent abstract workflow (Figure 3).

**[Abstraction of Files]:** We abstract files by the method explained in Section 4.1. If the user is accessing a file that has a corresponding file-cluster, we simply replace the being accessed file's filename in the log by the corresponding file-cluster. However, in some cases, such as newly created files, some files do not own corresponding file-clusters. In such cases, we first replace the file being accessed by the most similar file that does have a corresponding file-cluster.

**[Estimation of Abstract Tasks]:** We estimate the current abstract task being operated on by the user from the abstract files being accessed. We group the abstract files being accessed into a task, and then compare this task with tasks in the database to find the most similar task.

**[Estimation of Frequent Abstract Workflows]:** After estimation of the abstract task, we estimate the frequent abstract workflow. Because there are several abstract workflows that contain the estimated abstract task, we score each frequent abstract workflow in the database according to these three criteria.

- Degree of matching between $workflow\,being\,accessed$ and $workflow\,in\,database$
- Frequent occurrence score for $workflow\,in\,database$
- Number of possible tasks in $workflow\,in\,database$ for recommendation

**[Recommendation]:** The aim of this step is to recommend files and operations based on the estimated abstract task and frequent abstract workflow. If we know about the user's current abstract task and frequent abstract workflow, we can identify and recommend subsequent abstract tasks. However, a frequent

abstract workflow is a sequence of abstract tasks, while an abstract task is a set of pairs of operations and file-clusters. Each file-cluster contains several files. The problem is how to rank the recommendation candidates. Therefore, the proposed method extracts the last access time stamp of each file, ranking files belonging to the same file-cluster by most recent access time stamp.

## 5 Experiments


**Fig. 4.** Comparison of Workflow Characteristics

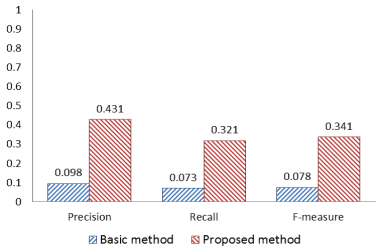


**Fig. 5.** Comparison of Precision, Recall, and F-measure for all Test Cases

The goal of our exteriment is to investigate the effectiveness of the features of our proposed concept, namely the abstraction of tasks and workflows. So we set up a basic method that did not use abstract tasks for comparison. The basic method simply extracts sequences of file operations as workflows and calculates those that are frequently used for recommendation.

The experimental data came from actual file-access logs provided by a commercial organization. There are 22 users in our log data. The record term is about 8 months and 9917 records and 1750 files are recorded in the log data. We split the log in a ratio of 70% to 30%. The first 70% of the log was for learning tasks and workflows and the other 30% was for evaluation. We divided the log for evaluation in terms of user IDs, using the same TGBW parameter to obtain workflows as used by the proposed method and created 151 test cases. Each test case is a workflow sequence. For each test case, the first 2 records from the beginning were input into our recommendation system to acquire a set of recommendation results. The remaining records in the test case were used as a correct answer set. By matching the examinees set and the results set, we obtained values for Precision, Recall, and F-measure. A comparison between the proposed method and the basic method was then made using the average values for all test cases.

We compared the workflows extracted by the two methods in bubble chart Figure 4. The horizontal axis represents the occurrence value (support-count value) for workflows, the vertical axis represents the number of file operations in

workflow sequences (sequence length), and the size of the bubbles represents the number of workflows. Note that basic method's bubbles are more concentrated in the lower left corner than proposed method's bubbles, which means that the workflows extracted by the basic method have smaller support-count values and shorter sequence lengths. Workflows with a small support-count value are more contingent and tend not to be reusable working patterns. In addition, workflows with small sequence lengths are undesirable because of the small amount of information available for the recommendation. Our proposed method can extract workflows with higher support-count values and greater sequence lengths than the basic method, which are more suitable for making recommendations.

We now describe the recommendation results for the evaluation experiment. Figure 5 shows the average Precision, Recall, and F-measure for all 151 test cases. Note that the proposed method performs much better than the basic method for all metrics. In particular, the F-measure score was improved from 0.078 to 0.341. The reason for this large difference is that the numbers of test cases that can be recommended are overwhelmingly different. Because the basic method can only recommend files that have been used in the past, only 25.8% of the test cases returned a result. On the other hand, because of our proposed method's abstract file operations, the proposed method can recommend files from similar file operations undertaken in the past. This enables more files to be available for recommendation. About 57.0% of the test cases returned results. We then focuses on the test cases that returned results. Our proposed method still performs considerably better than the basic method. The F-measure score was improved from 0.301 to 0.598. The reason for this is considered to be the quality of the workflows used in the recommendations. We therefore investigated the average support-count value for frequent abstract workflows that are used in the recommendations. The basic method's average support-count value for frequent abstract workflows is 4.833, while the proposed method's value is 20.186. We can conclude that the proposed method's workflows have higher support-count values and more suitable for making recommendations, which will improve the recommendation accuracy greatly.

## 6 Related Work

Okamoto et al. proposed a Web-page recommendation method using Web-access logs [4]. By extracting patterns in combinations of multiple attributes of the accessed pages, their method can also recommend new Web pages. Although each Web page is abstracted in their study, there is no concept of abstract tasks.

Tanaka et al. proposed a personalized document recommendation system via collaborative filtering [6]. In their system, documents viewed within a short period of time are considered to be related to the same working unit for the same purpose. They partition the access logs using the time gap between two records. We adopted this idea when extracting tasks from logs. Another file recommendation system was proposed by Lai et al. [3]. Their work proposes recommendation methods based on the knowledge-flow (KF) model. There are

two differences between their work and our proposed methods. First, the aims of file abstraction are different. In their work, files with similar topics (keywords) are grouped together. On the other side, we group files not depending on topics, but the purpose of use, which is more suitable for extracting meanful workflows. Second, their study groups similar files and then calculates KFs (workflows) directly, whereas our proposed method first groups similar files into tasks and then calculates the sequences of tasks as workflows. By introducing the concept of abstract task, our method can extract patterns with difference in order.

SUGOI [7] is for searching files using file access logs. SUGOI finds related files using file-access logs. In their study, a task is defined as the file set containing related files in simultaneous use. However, their method does not perform abstraction on tasks, which is the main difference from our study.

## 7 Conclusion and Future Work

In this paper, we propose a method to extract frequently used abstract-workflow patterns from the history, and to recommend files and operations by monitoring the current workflow of the target user. There are two points in our proposed method. First, our proposed method is able to extract general patterns, which are more suitable for making recommendations by abstracting such files. Second, the proposed method introduces abstract tasks to eliminate sequential relations inside tasks. The experiment results demonstrate that our proposed method is more suitable for recommendation. Consequently, the F-measure of the recommendation results was improved significantly from 0.301 to 0.598. In the future, we plan to consider a better algorithm for partitioning logs instead of simply using a fixed time. This might involve using information such as the frequency and type of operations.

## References

1. Feldman, S., Duhl, J., Marobella, J.R., Crawford, A.: The hidden costs of information work. IDC WHITE PAPER (March 2005)
2. Han, J., Kamber, M., Pei, J.: Data Mining: Concepts and Techniques, Second Edition. Morgan Kaufmann (2006)
3. Lai, C., Liu, D.: Integrating knowledge flow mining and collaborative filtering to support document recommendation. The Journal of Systems and Software pp. 2023–2037 (2009)
4. Okamoto, H., Yokota, H.: Access log based web page recommendation using multiple attributes of web pages (in japanese). Proc. WebDB Forum 2009 (Nov 2009)
5. Song, Q., Kawabata, T., Itoh, F., Watanabe, Y., Yokota, H.: Recommendation method for files and operations based on workows of abstract tasks from access log (in japanese). IPSJ (2013)
6. Taguchi, H., Sakojo, S., Iwata, M.: Personalized document recommendation for field engineers (in japanese). DEIM (2011)
7. Wu, Y., Otagiri, K., Watanabe, Y., Yokota, H.: A file search method based on intertask relationships derived from access frequency and rmc operations on files. DEXA pp. 364–378 (2011)